Gaussian Processes and CNNs: The utility of Prediction Variance

Joshua Send [1]

Abstract

Convolutional neural networks (CNNs) have long been used to achieve high classification accuracies on a huge variety of tasks. However, they struggle provide information beyond class probabilities, which have to be accepted at face value. Gaussian processes (GPs) are fully Bayesian constructs which can be used to predict both class probabilities and variances for those probabilities, indicating confidence. Using MNIST and related datasets, this paper investigates the effectiveness of combining GPs with CNNs, considering variance for interpretation of results, and as a tool for merging predictions from the base CNN and the GP. This merging process is shown to slightly increase test set accuracy in non-adversarial settings.

1. Introduction

Convolutional Neural Nets (CNNs) have seen a steep rise in use for a wide range of computer vision and machine learning tasks since Krizhevsky's success at the ImageNet challenge in 2012 (Krizhevsky et al., 2012). Given enough training data, they excel in domain specific applications and are efficient to train, but may confidently mis-predict classes, provide no feedback to users about variability of predictions, and have been shown to be vulnerable to adversarial attacks (Szegedy et al., 2013). On the other hand, Gaussian Processes are capable of classification and providing a variance about the prediction. This can be interpreted as a confidence in the probability of a prediction.

This work combines CNNs with Gaussian Processes (GPs), using the second to last layer of the network as the input to the GP. This combination takes advantage of neural networks' ability to approximate high dimensional data efficiently; GPs can then produce Bayesian uncertainty estimates along with predictions.

The goal of this work is to explore how useful the variance produced by a GP is, in terms of interpretability and enhancing prediction accuracy. Focus will be on image classification using the MNIST (LeCun et al., 1998) and N-MNIST (Basu et al., 2017) datasets, along with adversarially perturbed MNIST images.

[1]University of Cambridge. Correspondence to: Joshua Send <js2173@cam.ac.uk>.

2. Background

2.1. Convolutional Neural Nets

CNNs are composed of layers of neurons[1] with associated weights and activation functions. A CNN takes raw input (images) and reduces them to higher level feature representations, eventually producing some output (often a classification). Training is done via backpropagation, minimizing some loss function by adjusting the weights in the network. This is an $O(n)$ operation repeated some number of iterations.

To use CNNs for classification, the second to last layer is fully connected to a layer of m units, where m is the number of classes. The following *softmax* activation is then applied:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{k=m} e^{z_j}}$$

Combined with categorical-cross entropy loss during training, the resulting values between 0 and 1 from each of the m units represent the posterior distribution over the classes (Bridle, 1990). In other words, we obtain the probability of any class using a fully connected softmax layer. Note that this method reinforces large values and squashes small ones due to the exponentiation.

2.2. Gaussian Processes

One of GPs' key benefits is that little hyperparameter tuning is needed – since the model is fully Bayesian, parameters can be learned directly by maximizing marginal likelihood. However, traditional GPs are costly to train – covariance matrices are $O(n^2)$ in size, and inversion and decomposition operations have an $O(n^3)$ cost, where n is the number of training samples.

[1]A very coarse approximation to biological neurons

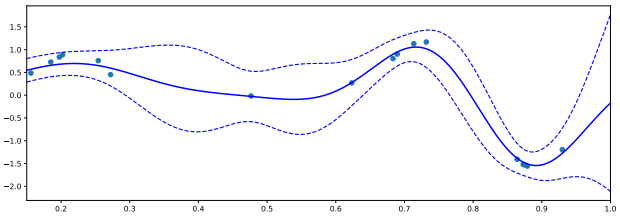


Figure 1. A simple one-dimensional GP regression. The dashed lines represent two standard deviations from the mean, and blue points are training data. Notice how the variance increases away from known datapoints.

To deal with the issue of scalability[2], various approximation methods have been developed. One approach that is used here is called Sparse Variational Gaussian Processes (Hensman et al., 2015). This classification approach effectively uses a set of *inducing points* optimized from the training data as an estimate of the full data set; these are then used to fit a GP.

Gaussian processes classification is described in Rasmussen and Williams (2006). This book also describes a variety of kernels, the choice of which strongly affects performance – some are explored in Section 5.

2.3. Adversarial Attacks

In this paper, an adversarially perturbed MNIST dataset is tested. The finite-gradient sign method (Goodfellow et al., 2014) (FGSM) is used to perform non-targeted attacks. This approach requires access to the trained model to maximize a loss function, performing gradient ascent away from correct classifications by adding small amounts of noise at each step. The amplitude of this noise is controlled by a parameter ϵ.

Other attacks, such as Jacobian-based saliency maps (Papernot et al., 2016) are not explored here, but the work could easily extended to include it.

3. Related Work

The key component of this work is using a fully Bayesian inference method to derive interpretable confidences, as well as variances for predictions. Some work exists along these lines, adapting CNNs to produce variances as well. One notable contribution is dropout-based approach (Gal & Ghahramani, 2016), using dropout at *test* time to sample a variety of predictions and computing a mean and variance from this.

Another approach is using fully Bayesian Neural Networks, an idea established in the 90s (MacKay David, 1992). These place a distribution over each parameter in the network, making inference and calculating the full posterior distribution theoretically possible but practically intractable. More recently there has been work towards this end that is back-propagation and GPU compatible (Blundell et al., 2015). Other work has applied similar ideas to recurrent neural networks (Zhu & Laptev, 2017).

Existing work combines CNNs and GPs. Bradshaw et al. (2017) investigate GPDNNs (Gaussian Process Deep Neural Nets) in a similar manner to this paper. However, they emphasize end-to-end trained models, comparing neural nets with softmax classification to ones with a GP replacing the softmax layer and retraining from scratch. In contrast, this work explores using pre-trained neural nets in combination with gaussian processes which are trained post-hoc.

In terms of resisting adversarial attacks, this work will evaluate the normal CNN and the GP model on perturbed MNIST images, showing modest gains and better interpretability in the GP. Most existing methods for protecting machine learning models from such attacks are either based on data augumentation (which does not scale), or training multiple networks with high overhead. Recently alternative approaches have been proposed, such as using GANs (Samangouei et al., 2018)(Zantedeschi et al., 2017).

4. Implementation

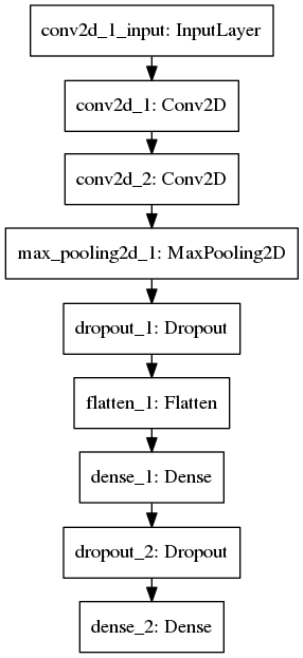


Figure 2. The structure of the CNN used throughput this paper. The final fully connected layer is *softmax* activated while the previous layers are *relu* activated.

[2]The training data here is a 60000x128 vector which is formed into a square matrix, far exceeding what could fit into memory

The core libraries used were Keras (Chollet et al., 2015) with a TensorFlow[3] backend for the MNIST CNN. GPFlow (Matthews et al., 2017) was used to implement the gaussian process model due to its support for large, high dimensional training data[4]. Scipy's machine learning toolkit was also tested but was found to be unsuitable.

The models tested are

1. Keras MNIST CNN trained on 60,000 training images.

2. 3 different GPs trained features extracted from the CNNs second to last layer.

3. A hybridized model merging CNN and GP predictions, motivated and developed in Section 5.5.

5. MNIST

This section focuses on the MNIST test data set, consisting of 10,000 examples, each 28x28 pixels.

5.1. Accuracy of and Motivation of Models

Figure 3 shows the test-set accuracy of the CNN, along with three different Gaussian Processes: one using a *Polynomial* kernel, one using the *Matern12* kernel, and the last a *Matern32*Linear* kernel (referred to as *combined* kernel from now on). For explanation of the standalone kernels please refer to Rasmussen & Williams (2006).

These kernels were chosen out of 17 tested single and combined kernels. It will be shown later that the *Polynomial* kernel performs well in adversarial situations, as does *Matern12*, while the *combined* kernel appears to inherit properties from its two components: *Linear* kernels exhibit strong performance on non-adverserial datasets, while *Matern32* performs worse across the board but is most effective at merging with CNN predictions. This will exploited in Section 5.5.

While exploring the GP design space, many configurations were tested. The most influential by far was the number of inducing points used. All kernels in this paper were created using every 25th data point as a possible inducing point and allowed to be automatically optimized during training. Minibatch sizes were set at 8000. A Gaussian noise kernel was added to all kernels with constant variance 0.1, though its inclusion nor its variance have much impact on results. Allowing the GP to optimize a lengthscale parameter per input dimension did not improve performance but increased training time and was left out.

[3]https://www.tensorflow.org/

[4]GPFlow implements sparse variational GPs described in Section 2.2 as well as batched training

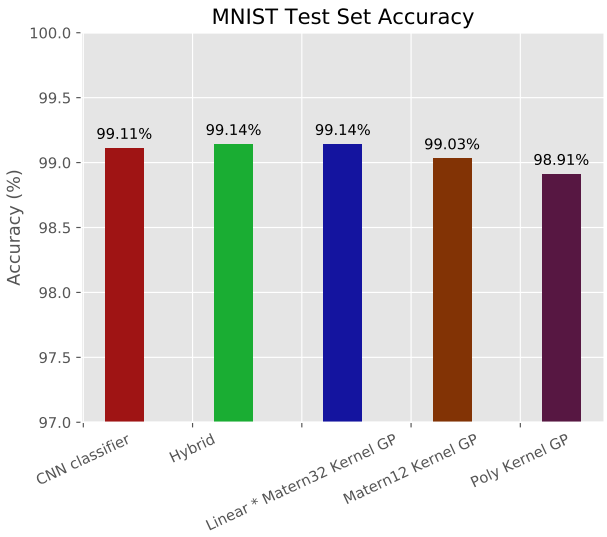


Figure 3. Accuracy across the different models presented throughout this paper. The Hybrid model will be developed in Section 5.5. The models perform comparably, except the *Polynomial* kernel which is marginally worse.

5.2. Distribution of Variance

The previous section simply compares the performance of *softmax* versus a GP as a feature classifier. The goal of this paper is to examine the utility of prediction variance, which the CNN does not provide. Figure 4 examines the distribution of variance for correctly, and incorrectly classified examples using the *combined* kernel.

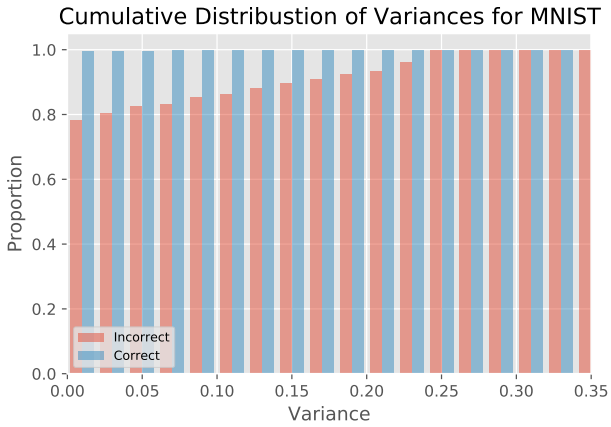


Figure 4. Distribution of Variances across correctly classified examples. 95% of these classifications occur with a variance of 0.02 or less.

This plot clearly shows that the variance of incorrectly predicted classes is in general higher than when correctly classified. One might expect correct classifications are on average confidently predicted. Indeed, this is supported here,

showing inputs that are incorrectly classified have a lower confidence (higher variance).

This is a useful property when discussing interpretability: by presenting the variance to human users, a judgement can be made about how much to trust the predicted class, or whether consideration should be given to other predictions that have lower probability but are within, say, a standard deviation of the most likely class.

5.3. SVGP Variance

The GP implementation used in this paper is the sparse variational gaussian process (SVGP) implemented in GPFlow. It is worth double checking that the variance being predicted makes sense and is providing additional information.

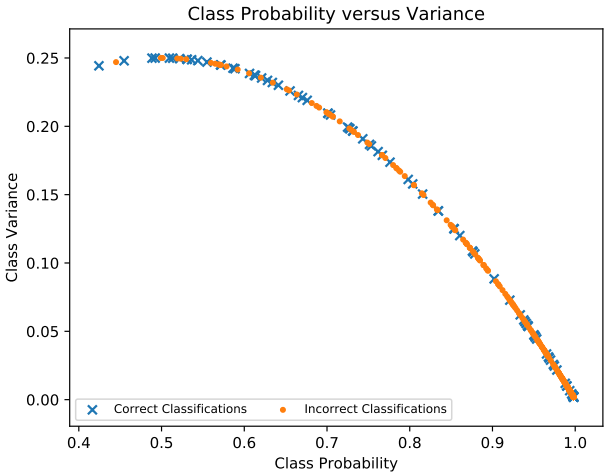


Figure 5. Plotting class probability versus variance on the standard MNIST dataset. They are totally correlated and thus one completely determines the other.

The above figure shows that, actually, with this technique and implementation, the class probability completely determines the predicted variance. In fact, a peek into the implementation reveals that variance σ^2 is calculated from the predicted mean p as $\sigma^2 = p - p^2$.

Note that this observation does not invalidate the usefulness of variance – it simply says that there is no point getting both mean and variance from the SVGP and analyzing them separately. Calculating the variance and in turn standard deviation is still a useful indicator for explaining results, and later it shall be used to merge *softmax* and GP predictions.

A human operator that did not have this information may try to interpret the class probabilities and variances independently. While not harmful, it could certainly lead to confusion and doubt about the system being presented to them.

5.4. Examining Misclassifications

It is worth further examining the misclassifications on the MNIST dataset, for both types of models. I again focus on results from the *combined* kernel as representative of all three GP models.

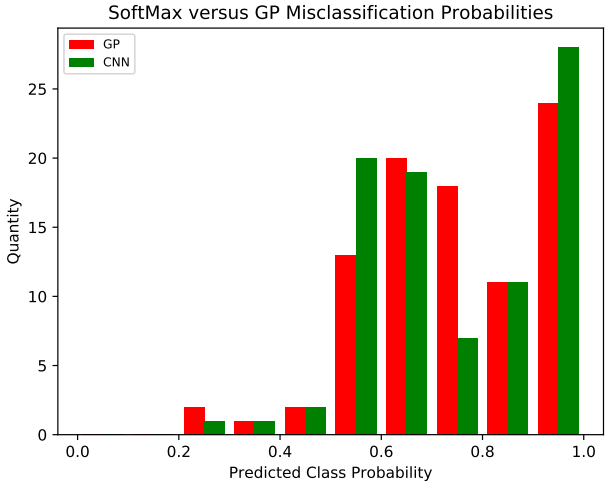


Figure 6. Misclassification confidences for CNN and GP with the *combined* kernel.

We see that the CNN is slightly more confident when mispredicting than the GP. This can be seen in Figure 6 and Table 1. From this we can conclude that we should put more trust into the GP's prediction probabilities than the CNN's, a fact used later.

Table 1. Mean and Standard Deviation of probability of incorrect predicted classes for the *combined* kernel GP and the CNN.

STATISTIC	GP	CNN
MEAN	74.5%	75.3%
STANDARD DEVIATION	17.7%	18.5%

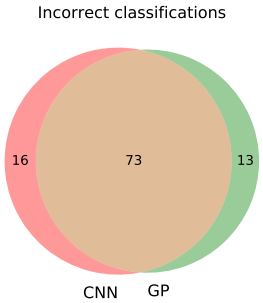
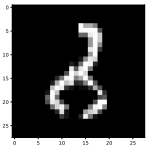


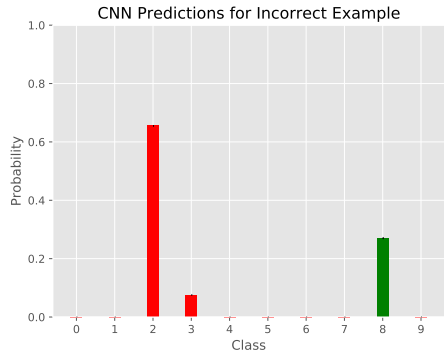
Figure 7. Overlap in misclassifications of the CNN and the *combined* GP model.

The Venn diagram in Figure 7 illustrates that both models

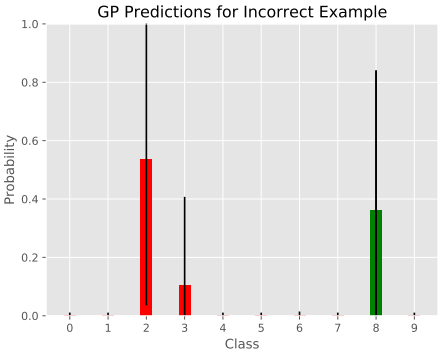
largely fail on the same 73 test images. These are digits such as the one in Figure 8, and can be seen as the truly difficult or ambiguous images. Nothing much can be done about them.



(a) Hard Digit



(b) CNN Prediction



(c) GP Predictions with 1 standard deviation

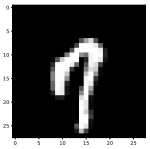
Figure 8. An example of a truly difficult image that both models fail on. The green bar represents the true class, while the tallest red one is the predicted class.

However, more interesting is the non-overlapping region of the Venn diagram. From Figure 9 we see there is hope for boosting accuracy by combining the results from the CNN and the GP: the true class, in green, has a high probability but is not the highest. This inspires the hybridized model in the next section.

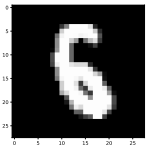
5.5. Hybridization

Because we have a (small) ensemble of classifiers when using both the CNN and a GP, a disagreement amounts to an extra bit of information. We can try to choose the best result, steered by the variance predicted by the GP.

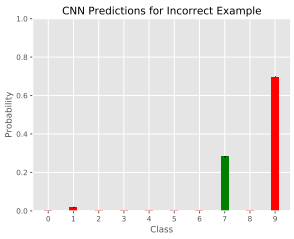
The driving principle in this heuristic is that we trust the Gaussian Process model more than the CNN, since the CNN gives no indicator of how reliable its result is and more confidently mis-predicts results. It is also more likely to fail under adversarial conditions (Section 7).



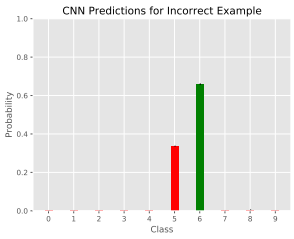
(a) CNN Misclassified



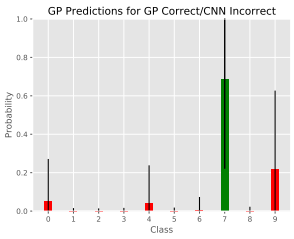
(d) GP Misclassified



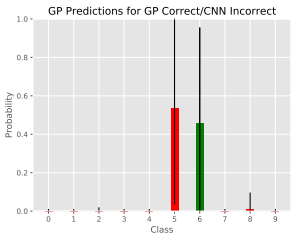
(b) CNN Wrong Predictions



(e) CNN Right Predictions



(c) GP Right Predictions



(f) GP Wrong Predictions.

Figure 9. Examining cases of misclassifications. The left column corresponds to one of the 16 examples the CNN classified incorrectly, while the right column stems from the GP's incorrect classifications. With the right heuristic these case, and others, may be salvageable.

5.5.1. Algorithm

When merging CNN and GP predictions, we default to the GP prediction, only choosing the CNN's result if the following conditions hold:

1. The CNN's top prediction is within d standard deviations of the corresponding GP's probability (calculated using corresponding GP variance). This ensures that the GP agrees the CNNs prediction is plausible.

2. The CNN's top prediction is within d standard deviations of the top GP's prediction (calculated using top GP variance).

3. The GP's top prediction not d or more standard deviations from the GP's corresponding prediction.

The first condition pulls the GP accuracy toward the CNN to either increase or decrease total accuracy. The latter two conditions mitigate worsened accuracy to an extent, ensuring that the GP's decision is accepted when it is very confident

compared to the CNN and that the CNN's probability isn't too low, even if plausible.

5.5.2. Discussion

There are some points to discuss regarding this hybridization.

Firstly, when accepting a CNN prediction, we lose a meaningful variance and the resulting interpretability a human might utilize. However this could easily be fixed by flagging the result as uncertain due to disagreement, as well as presenting the GP predictions.

Secondly, in this situation some kernels boost accuracy more than others. I believe this stems from mis-classifying different images from the CNN. Through experimentation the *Matern32*, and in turn the combined *Linear*Matern32*, kernels were found to provide the best results.

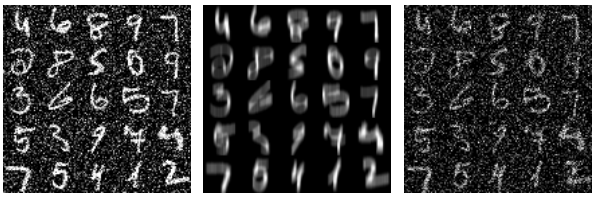
Thirdly, other CNN acceptance criteria are possible. These three conditions and subsets of them were tested before settling on their intersection, as it offered reasonable improvements across all experiments.

Finally, if the CNN and the GP output are saved anyway, this process is relatively cheap, being $O(m)$ in the number of classes m.

From here on I also present hybridized model results using the *combined* kernel with an acceptance criteria of $d = 0.2$ standard deviations.

6. n-MNIST

The 'noisy'-MNIST dataset stems from Basu et al. (2017). The authors provide three transformations of the MNIST images: added white noise, motion blurred, and lowered contrast and white noise.



(a) (b) (c)

Figure 10. Added white noise, motion blurred, and low contrast + white noise samples. Images from http://csc.lsu.edu/~saikat/n-mnist/

6.1. Accuracy

Figure 11 shows how each model performs across the different n-MNIST datasets. The hybrid model consistently outdoes the standard *Linear * Matern32* kernel by a small amount. It is reassuring to note that the heuristic devised is not harming performance, and boosting it by 0.3% in the best case.

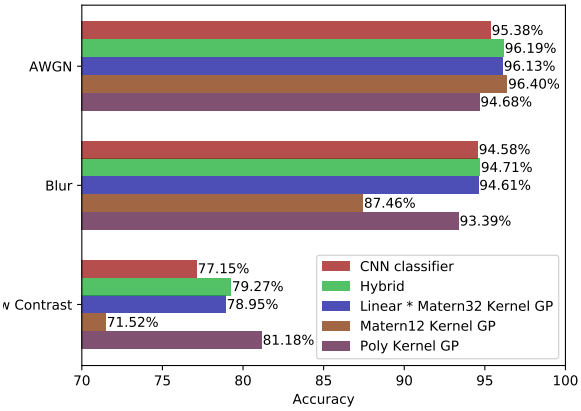


Figure 11. Performance across all models and n-MNIST data sets.

Most kernels outperform the CNN excepting the *Matern12* kernel. This kernel allows quite a lot of movement, i.e. the covariance between close data points is low, so it is possible it overfits the training data and suffers in very different situations. However, drawing strong conclusions would require more mathematical analysis.

From this experiment, it is quite evident that each kernel has some strengths, and none of them outperform the others across the board. This could be the basis of larger ensemble methods, and an idea discussed in Section 8.

6.2. Distribution of Variance

The distribution of variances on the various n-MNIST test sets are as expected, showing the correct examples having much lower variances than the incorrect examples.

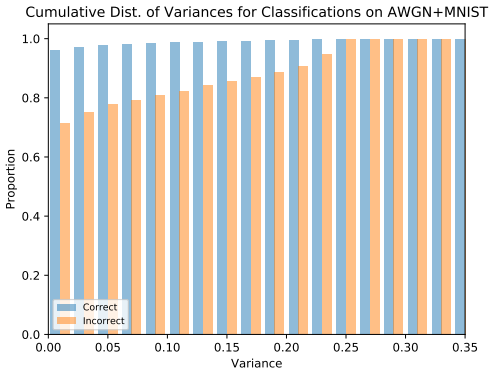
Figure 12 also shows that as we analyze datasets that result in lower overall accuracy, the distribution of correct variances approaches the incorrect one.

7. Adverserial Attacks

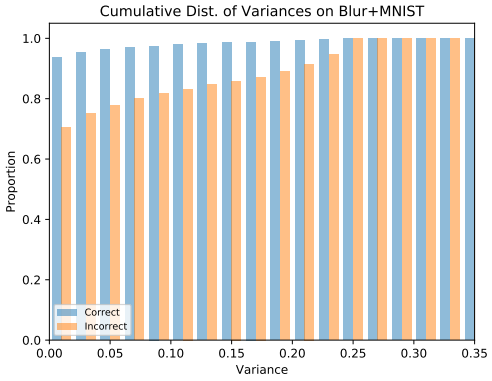
Adverserial attacks are described in Section 2.3. FGSM is used to perturb the 10,000 MNIST images using the trained CNN as the target model.

Using an $\epsilon = 0.2$, all the models show large performance degradations from the 99% accuracy presented in Section 5.1.

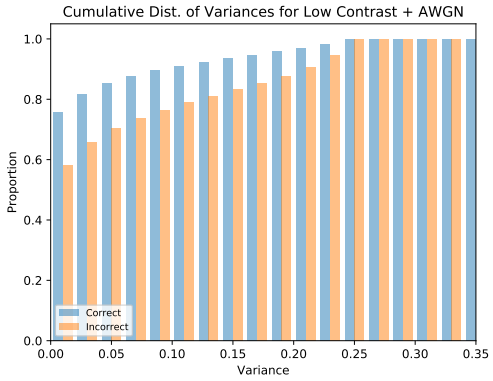
Unfortunately, Figure 13 shows that the hybridized model actually performs slightly worse than the *Linear * Mater32* kernel GP. This is because the CNN is often quite confident



(a) With white noise



(b) With motion blur



(c) Low contrast and with white noise

Figure 12. Variances across the n-MNIST dataset

in its predictions, and the extra conditions to revert to GP predictions are not strong enough to combat this. From experimentation it seems that making stronger conditions or reducing the acceptance parameter d further would prevent this loss, at the expense of smaller gains on the other data sets.

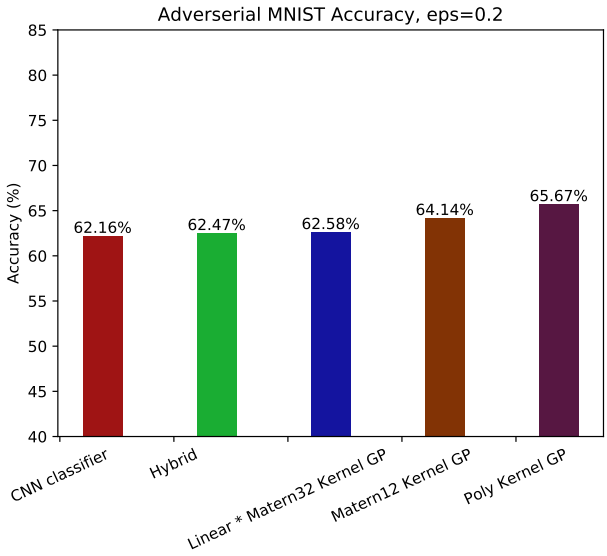


Figure 13. Accuracies on FGSM perturbed MNIST, $\epsilon = 0.2$

7.1. Varying ϵ

Previous work by Bradshaw et al. (2017) showed that for GPDNNs, which have GP's trained end to end with CNNs from scratch, targeting the plain CNN architecture does not harm the GPDNN much. This conclusion is rather obvious since they learn different structure. Vice versa, their CNN is resistant to attacks on the GPDNN.

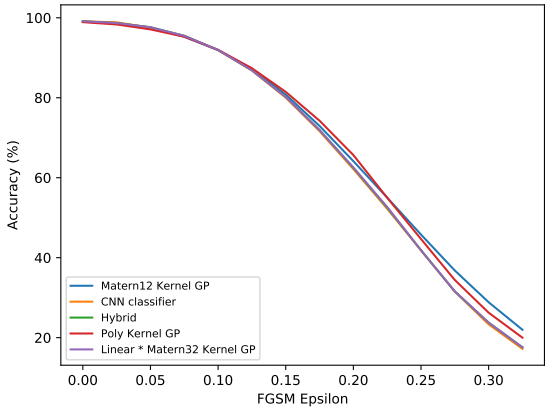


Figure 14. Classification Accuracies on FGSM attacked MNIST images across varying ϵ

Here, I show that substituting the *softmax* layer with a GP and not training from scratch yields some accuracy improvements, if the right kernel is used. However, as Figure 14 shows, even in the best case with *Polynomial* or *Matern12* kernels, at most 5-10% boosts can be expected as ϵ becomes large. The *combined* and hybrid models perform as badly

as the CNN.

8. Discussion

The best use of variance is presented to a human user as standard deviations along with class probabilities. It would hardly be far fetched to imagine a system that flags uncertain results based on variance for further inspection. This is especially useful in situations where false negatives are costly, such as in the medical field.

As noted in Section 5.3, in this implementation variance is actually just as informative as class probability. This does not mean the variance is useless information since humans can easily reason about and visualize standard deviations and thus uncertainty.

An alternative use of variance, the hybridizing approach, was found to provide incremental gains over the *Linear * Mater32* kernel. Unfortunately, it also slightly worsened performance in adversarial situations. It is possible that more consistent accuracy gains are within reach with more sophisticated methods. One obvious approach is to further refine the acceptance criteria described in Section 5.5.1, or experiment with different acceptance parameter values, or even using 3 different parameters, one for each part of the criterion. However, none of these are likely to lead to particularly novel algorithms or dramatically improved results.

As a positive, creating the hybridization is almost free: most of the CNN needs to be evaluated to extract features to feed into the GP. For instance, in TensorFlow, executing the entire CNN and saving an intermediate value incurs almost no cost. If the gaussian process is going to be executed in $O(n^2)$ anyway, the hybridization is only $O(m)$ (for m classes) more expensive.

Whether training the GP at all is worth it is another question: even with sparse variational implementations, GPs still do not scale as well as CNNs, and this investigation has revealed that classification accuracy is just on par with CNNs. In situations where interpretation is important, it may be worth re-evaluating the entire training set to train the GP. However, if interpretability is not a concern, I argue using a GP is not useful. Additionally, if evaluation speed is important, a small CNN can be made to execute quickly and at a constant speed regardless of training data site – GPs predict quadratically in the number of training samples.

9. Future Work

9.1. Ensembles

Hybridization between a single GP and a CNN ultimately provides unimpressive results. A more interesting idea could be take the entire process a step further: hybridize based on a multitude of kernels, effectively forming an ensemble.

As mentioned before, each kernel appears to have different strengths, and it may be possible to come up with a heuristic for merging many predictions. Part of such an investigation would be to explore the role of variance. More heuristics could be developed, or reinforcement learning could be used to decide on an optimal decision strategy, though this would be biased toward its training data and be difficult to generalize.

9.2. Automatic Kernel Building

It was shown in Section 7 that kernels have varying adversarial resistance. Another investigative avenue could thus be automatically creating the best kernel for both a data set and adversarial resistance. Automatic kernel choice is already an active research area (for example (Abdessalem et al., 2017), (Duvenaud, 2014)), but without the adversarial effects taking into account. Further, it might be interesting to consider creating kernels that can effectively be ensembled together, i.e. err on maximally different parts of the same dataset.

10. Conclusion

This paper examined the utility of variance in classification. To do this, a CNN was trained on the MNIST dataset, and compared to various gaussian processes on a variety of datasets. The main benefit of variance is interpretability: having access to an uncertainty in a prediction means consumers of classifications can make further judgements about results and don't need to accept predictions at face value.

This paper also presented the idea of merging predictions from the base CNN and the GP, using variance as a guide for choosing one decision of the other. A heuristic was presented which achieved slightly increased test set accuracy on non-adversarial examples. Overall however, a simple two-way merge does not yield impressive results, and a further investigation could be conducted into ensembling multiple different gaussian processes, also steered by prediction variances.

10.1. Software and Data

Code, results, and this report can be found at:

`https://github.com/flyingsilverfin/CNN_GP_MNIST`

Please note that intermediate results are not saved but can be recomputed, and that some of the configurations are specific to the development machine.

References

Abdessalem, Anis Ben, Dervilis, Nikolaos, Wagg, David J, and Worden, Keith. Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Frontiers in Built Environment*, 3:52, 2017.

Basu, Saikat, Karki, Manohar, Ganguly, Sangram, DiBiano, Robert, Mukhopadhyay, Supratik, Gayaka, Shreekant, Kannan, Rajgopal, and Nemani, Ramakrishna. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. *Neural Processing Letters*, 45(3):855–867, 2017.

Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Bradshaw, John, Matthews, Alexander G. de G., and Ghahramani, Zoubin. Adversarial Examples, Uncertainty, and Transfer Testing Robustness in Gaussian Process Hybrid Deep Networks. pp. 1–33, 2017. URL http://arxiv.org/abs/1707.02476.

Bridle, John S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing*, pp. 227–236. Springer, 1990.

Chollet, François et al. Keras. https://github.com/keras-team/keras, 2015.

Duvenaud, David. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Gal, Yarin and Ghahramani, Zoubin. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Goodfellow, Ian J, Shlens, Jonathon, and Szegedy, Christian. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Hensman, James, Matthews, Alexander G de G, and Ghahramani, Zoubin. Scalable variational gaussian process classification. 2015.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

MacKay David, JC. A practical bayesian framework for backprop networks. *Neural computation*, 1992.

Matthews, Alexander G. de G., van der Wilk, Mark, Nickson, Tom, Fujii, Keisuke., Boukouvalas, Alexis, León-Villagrá, Pablo, Ghahramani, Zoubin, and Hensman, James. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40): 1–6, apr 2017. URL http://jmlr.org/papers/v18/16-537.html.

Papernot, Nicolas, McDaniel, Patrick, Jha, Somesh, Fredrikson, Matt, Celik, Z Berkay, and Swami, Ananthram. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016.

Rasmussen, Carl Edward and Williams, Christopher KI. *Gaussian process for machine learning*. MIT press, 2006.

Samangouei, Pouya, Kabkab, Maya, and Chellappa, Rama. Defense-gan: Protecting classifiers against adversarial attacks using generative models. 2018.

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian, and Fergus, Rob. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Zantedeschi, Valentina, Nicolae, Maria-Irina, and Rawat, Ambrish. Efficient defenses against adversarial attacks. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 39–49. ACM, 2017.

Zhu, Lingxue and Laptev, Nikolay. Deep and confident prediction for time series at uber. In *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*, pp. 103–110. IEEE, 2017.